06

Estructura proyecto

Transcripción

[00:00] Bien. Ya tenemos nuestro proyecto generado. Ahora vamos a explorar qué es lo que hay aquí dentro. Para eso necesitamos un editor de código, en este caso yo uso IntelliJ, uno de mis favoritos, y vamos a abrir proyecto que hemos generado justo ahora. Vamos a descargar y vemos aquí está nuestro API.

[00:19] Damos abrir, le vamos a dar que sí, que confía en el proyecto porque lo ha generado el Spring, y vemos cómo aquí me sale mensaje de indexando dependencia, vamos a agrandar esto y vemos como IngelliJ automáticamente comienza ya a descargar las dependencias.

[00:39] Por ejemplo aquí en Maven si nos damos cuenta aquí en API, tenemos todo un lifecycle de las operaciones de Maven pues el gestor de dependencias que hemos decidido utilizar para esto es obviamente Maven.

[00:53] Vemos aquí que está sincronizando el API, está descargando las dependencias en este momento. Normalmente puede tomar algunos minutos, dependiendo cuántas dependencias tenga tu proyecto. Vamos a hacer una pausa en la grabación aquí y continuar en este punto cuando termine de descargar la dependencia.

[01:17] Bien. Vemos que ya la descarga de dependencias ha terminado. Casi dos minutos en este caso. Y aquí ya tenemos una sección de plugins y una sección de dependencias. Aquí lo que vamos a ver es verificar si las dependencias que nosotros seleccionamos al momento que creamos el proyecto están aquí.

[01:38] Por ejemplo, tenemos el spring-boot-starter-web, que es el que elegimos el Spring Web, el DevTools, que también es el que elegimos y la librería Lombok, que si bien no es propia de Spring, tenemos que ser muy claros en esto. Lombok no es propia de Spring pero Spring también lo integra de una forma muy fácil en el Initializr, porque es muy usada actualmente.

[02:01] Entonces Spring considera que es simple de implementar con esto. Ahora también tenemos el spring-boot-starter-test, que ustedes preguntarán ¿nosotros no elegimos esto? Okay, no lo elegimos pero por buena práctica Spring ya nos da una librería para poder crear tests unitarios en nuestro proyecto.

[02:19] Recuerden, todo código debe ser probado, testeado. Tenemos en los plugin el clean, compiler, deploy, install. Todos esos son plugins propios de Maven para los comandos en Maven que deseamos usar, eso lo podemos ver después, no se preocupen y finalmente, bueno, tenemos ya todo lo que necesitamos para explorar el proyecto.

[02:39] Vamos a cerrar nuestra pestaña de Maven que está de este costado porque no vamos a explorar más por ahora. Y vemos qué tenemos por aquí por el lado de nuestro API, qué es lo que ha generado el Spring Initializr. Tenemos el archivo pom.xml, los que ya están familiarizados con Maven ya conocen este archivo. Tenemos los esquemas propios de Maven de donde él está heredando la estructura de este archivo de xml.

[03:06] El parent, que tenemos aquí que es el spring-boot-starter-parent. ¿Qué significa esto de aquí? Si ustedes se dan cuenta al, usar Spring Boot nosotros no tenemos ninguna dependencia que sea, explícitamente Spring Boot como dependencia.

[03:22] Por ejemplo en la sección de dependencias tenemos el web que ya lo vimos, el DevTools, Lombok y test. Y ahí termina nuestra relación de dependencia. Ustedes dirán: ¿Dónde está Spring Boot? Bueno, Spring Boot

viene del parent. El parent es como que el padre en el como que el padre de este pom. Por lo tanto al especificar el parent, Spring Boot, esto quiere decir van a heredar todas las características que existen en el pom de Spring Boot.

[03:54] Actualmente en Spring Boot ya tiene todo configurado. Tiene un pom entero con todas las dependencias que necesita para ser configurado, para ser ejecutado, por lo tanto tú no tienes que repetir todo ese código aquí. Por ejemplo, si tú vas a tener una aplicación que va a necesitar todas estas dependencias, y va a estar dentro de tu misma empresa, tu misma jerarquía, tu mismo artefacto.

[04:19] Simplemente puedes definir este artefacto de aquí como el parent y no vas a tener que copiar nuevamente todas las recetas, digámoslo de esa forma, toda la receta de dependencias para este nuevo proyecto. ¿Qué más tenemos aquí? Bueno, tenemos la versión de Java, que tenemos 17, la descripción, el nombre, la versión versión 0.0.1, snapshot, la lista de dependencias que ya las vimos y los plugins que también ya los hemos visto.

[04:47] En este caso estamos excluyendo Lombok de la lista de plugins para lo que es la generación del artefacto aquí. ¿Qué más tenemos por aquí? Tenemos los wrappers de Maven, para poder ejecutar Maven automáticamente vía wrapper. Tenemos el GitIgnore, que es simplemente para no commitear los archivos que vayamos generando. Perdón, los archivos autogenerados por el IDE, por ejemplo el idea.

[05:18] Si damos doble clic aquí, vamos a ver pues que hay muchos activos autogenerados como los targets, donde se guardan los artefactos generados por el comando install por ejemplo. En fin, es una infidencia que podemos ignorar aquí, el vscode, si usas Visual Studio Code, aquí lo puedes agregar tranquilamente .idea para ignorar todos los archivos de este folder, idea, porque no es bueno compartirlo.

[05:45] ¿Qué más tenemos aquí? Finalmente el source, la carpeta source. Entramos aquí y tenemos main y tenemos test. En main tenemos la parte de Java, med.voll.api. Y eso es lo que nosotros especificamos como nombre de paquete si ustedes lo recuerdan, el nombre del paquete allá en Spring Initializr. Y ya tenemos una clase Java llamada ApiApplication.

[06:10] Este contiene un método main, que es el que se usa en Spring Boot para ejecutar vía esta aplicación como tal. No se preocupen, vamos a ejecutarla en un momento más. Primero hay que explorar a ver qué es lo que tenemos por aquí. En la parte de recursos o resources tenemos la carpeta static, que es para guardar archivos, JavaScript o .CSS para hojas de estilos en caso que estés creando una aplicación web.

[06:39] Como no lo vamos a hacer, no tenemos nada que guardar aquí, entonces puedes tranquilamente saltarlo. La parte de templates es para guardar todo lo que son páginas HTML, todos los template.html. Tampoco lo vamos a utilizar, entonces no se preocupen, no hay ningún problema.

[06:55] Y la application.properties es para almacenar propiedades del archivo. Por ejemplo, si tú quieres en tu código especificar una versión, alguna característica predeterminada nos puede ser útil crear en un archivo properties, no te preocupes. Igualmente vamos a ver todo esto muy a detalle a lo largo del curso porque es muy usado, no se preocupen por nada.

[07:20] Y cerramos ya la carpeta main y entramos a test. ¿Qué tenemos en test? Tenemos la parte de Java. Es una estructura igual a la que tenemos en main pero tenemos una clase de test aquí ya creada, en la cual Spring nos dice: “bueno, esta clase de test es la que viene por defecto. Es un ejemplo para que tú implementes las próximas”.

[07:43] No hay mucho más que decir aquí por ahora sobre la estructura del proyecto, vemos que ya el comando Maven está ejecutado, tenemos las dependencias tenemos la estructura creada, es un típico proyecto Maven, no

hay nada nuevo aquí hasta el momento. En el próximo video vamos a ejecutar ya este proyecto creado.